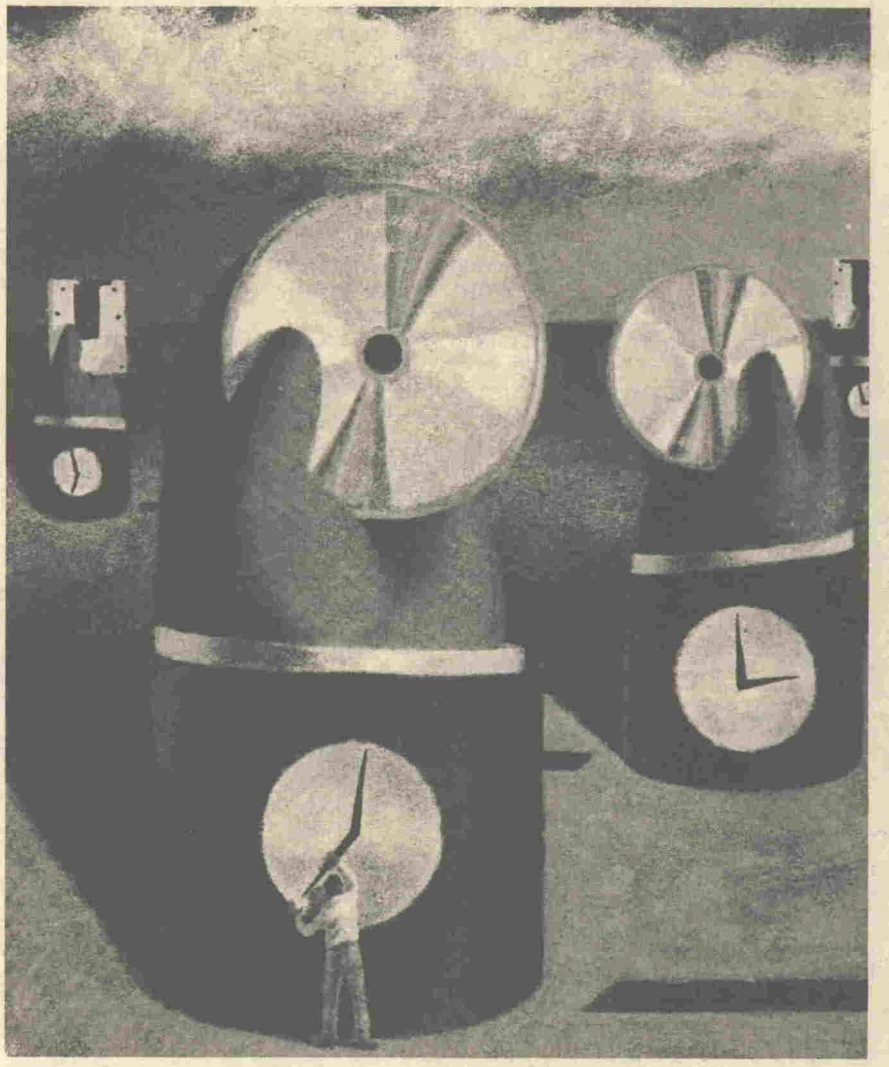
อาจหาญ สัตยารักษ์

SOFT MAIL

ทางเลือกใหม่ในการจัดการฐานข้อมูลดีเบส (3)

ศึกษาการเขียนโปรแกรมจัดการข้อมูลดีเบสกันต่อในคำสั่งนับ, รวม และเรียงลำดับ



มาศึกษาการจัดการกับฐานข้อมูลดีเบสด้วยปาสคาลต่อกันอีกเป็นตอนที่ 3 โดยจะเขียนเพิ่มในเรื่องการใช้คำสั่งนับ (count), การรวมข้อมูลชนิดตัวเลข (sum), การเรียงลำดับข้อมูล (sort) และการแก้ไขชื่อของเขตข้อมูล

โปรแกรมสำหรับตอนนี้อยู่ในไฟล์ชื่อ dbase2.pas ผู้อ่านที่ไม่มีเวลาคีย์โปรแกรมก็ติดต่อทางซอฟต์เมล์ได้เช่นเคย

โดยมีโปรแกรมที่สองอีกหนึ่งโปรแกรม (รูปที่ 7) ซึ่งเป็นการใช้หน่วยความจำที่เรียกว่าฮีพ (heap) มาช่วยในการทำงานพร้อมกับเขียนรูปแบบของไฟล์ใหม่ โดยกำหนดเป็นแบบเรคอร์ด (record) แทนแบบที่กล่าวมาแล้วทั้งหมด ซึ่งจะช่วยให้การเรียงลำดับข้อมูลใช้เวลาน้อยกว่า แต่มีข้อเสียคือจะใช้ได้กับแฟ้มข้อมูลดีเบสที่มีโครงสร้างแฟ้มเหมือนกันเท่านั้น ไฟล์ในซอฟต์เมล์ใช้ชื่อว่า sort.pas

โดยมีแฟ้มข้อมูลชื่อ vr.dbf ซึ่งมีโครงสร้างข้อมูลแบบเดียวกับแฟ้ม test.dbf ที่กล่าวในตอนแรกมาเป็นไฟล์ตัวอย่างในการศึกษาเปรียบเทียบการเรียงลำดับข้อมูลทั้งสองแบบว่าใช้เวลาต่างกันอย่างไร หากผู้อ่านไม่ได้สั่งซื้อซอฟต์เมล์ก็ลองสร้างไฟล์ขึ้นมาเอง หรือจะดัดแปลงโครงสร้างตัวอย่างใหม่ก็ได้

ตัวแปรที่เพิ่มขึ้น

คราวนี้มีตัวแปรที่ส่วนหัวเพิ่มขึ้นอีกเพียง 2 ตัวแปรเท่านั้น คือ

dbaseNum:real; สำหรับเก็บข้อมูลตัวเลขในเขตข้อมูลที่ตรวจสอบ

dbaseSTR:string; เก็บข้อมูลที่เป็นข้อความในเขตข้อมูลที่ระบุ

ขอให้พิมพ์เพิ่มเติมที่ส่วนต้นของโปรแกรม เพราะจะนำไปใช้ในการนับ, รวม และการเรียงลำดับข้อมูลตามชนิดของข้อมูล

ชุดคำสั่งช่วยงานเพิ่มเติม

เราจำเป็นต้องมีชุดคำสั่งเพิ่มเติมเพื่อให้การทำงานมีความสมบูรณ์ขึ้น โดยมี 1 ฟังก์ชัน และ 1 ชุดคำสั่ง ดังที่ปรากฏในรูปที่ 1 คือ

ฟังก์ชัน Cut สำหรับตัดอักขระรหัสแอสกี้ 0, 32, 160 ออกไปจากข้อความที่กำหนด โดยตัดเฉพาะที่อยู่หน้าหรือหลังรหัสแอสกี้อื่น ๆ เท่านั้น

ชุดคำสั่ง FindData สำหรับนำข้อมูลที่อยู่ในเขตข้อมูลที่กำหนดมาเก็บในตัวแปร dbasenum และ dbaseSTR

โดยขออธิบายดังนี้

ฟังก์ชัน Cut ต้องส่งตัวแปรข้อความ (content) เข้ามาและจะได้ตัวแปรข้อความกลับคืนไปเมื่อออกจากฟังก์ชันนี้

ในช่วงวนรอบ repeat ครั้งแรกเป็นการตัดรหัสแอสกี้ 0, 32 และ 160 ออกจากด้านหน้าของข้อความที่กำหนด โดยหยุดการวนรอบในกรณีที่อักขระตัวแรกของข้อความนั้น ๆ ไม่ใช่รหัสแอสกี้ 0, 32 และ 160 หรือเมื่อข้อความนี้ถูกตัดทิ้งหมด (content [0] := #0)

ช่วงการวนรอบ repeat ถัดมาเป็นการตัดรหัสแอสกี้ 0, 32 และ 160 ออกจากด้านท้ายของข้อความ โดยหยุดการวนรอบนี้เมื่อได้อักขระตัวท้ายสุด (content[x]) ไม่ตรงกับรหัสแอสกี้ 0, 32 และ 160 หรือเมื่อค่า x=0 ซึ่งแสดงว่าข้อความนี้ถูกตัดทิ้งหมดเลย

จบด้วยการกำหนดให้ฟังก์ชัน Cut:=content เพื่อนำคืนกลับออกไปใช้งานต่อไป

ชุดคำสั่ง FindData เมื่อการตรวจข้อมูลที่อยู่ในเขตข้อมูลที่กำหนด ทำงานดังนี้

ก่อนที่จะเรียกใช้ชุดคำสั่งนี้เราจะต้องเรียกใช้ชุดคำสั่ง FindField เพื่อหาตัวแปร NoFiled มาก่อนแล้วซึ่งจะทำให้ได้ตัวแปร xstart ด้วย เราต้องกำหนดหมายเลขของระเบียนข้อมูล (R) เข้ามาใช้ในชุดคำสั่งนี้เพื่อนำไปหาจุดเริ่มต้นของข้อมูลของเขตข้อมูลนั้นจาก x:=(R-1)*Size of Record+Xstart;

การนำข้อมูลเข้าสู่ตัวแปร STT นั้นจะอยู่ในช่วง x ถึง x+widthOffield [Nofield]-1 ซึ่งจะทำให้ตัวแปร STT เป็นข้อมูลที่บรรจุในเขตข้อมูลที่เราค้นหา

จบชุดคำสั่งนี้ด้วยการเปลี่ยนตัวแปร STT ไปเป็นตัวแปร DbaseNum และ DbaseSTR

```
FUNCTION CUT(Content:string):string;
var x:byte;
begin
   Repeat
     if Content[1] in [#0,#32,#160] then Delete(Content,1,1);
   until (Content[1] in [#1..#31,#33..#159,#161..#255]) or (Content[0]=#0);
  if Content[0]<>#0 then begin    x:=Length(Content);
   Repeat
     if Content[x] in [#0,#32,#160] then Delete(Content,x,1);
     Dec(x);
   until (Content[x] in [#1..#31,#33..#159,#161..#255]) or (x=0);
                      end;
   CUT:=Content;
end;



PROCEDURE FindData(N:word);
var x,s:word; STT:string;  ck:integer;
 begin
   x:=(N-1)*SizeOfRecord+Xstart; STT:=''; S:=x;  ck:=0;
   repeat
     STT:=STT+DataDbase[s];
     Inc(s); if DataDbase[s]=#0 then ck:=27;
   until (s>(x+WidthOfField[NoField]-1)) or (ck=27);
     Val(STT,DbaseNum,ck);  DbaseSTR:=STT;
end;
```

รูปที่ 1 ฟังก์ชันช่วยงานและชุดคำสั่งสำหรับหาข้อมูลในเขตข้อมูล

การเรียงลำดับข้อมูล

การเรียงลำดับข้อมูลเป็นสิ่งจำเป็นอย่างหนึ่งของการจัดการฐานข้อมูล ซึ่งในกรณีการดำเนินการด้วยดีเบสนั้นจะมีการอ่านและเขียนไฟล์แบบแผ่นดิสค์ตลอดเวลา สมมติว่าให้เก็บข้อมูลที่เรียงลำดับในแฟ้มชื่อ test2 ดีเบสก็จะดำเนินการดังนี้

1. ก็อบปี้แฟ้มต้นฉบับไปสำรองไว้ในชื่อ test2.w44 ก่อน เพื่อกันเนื้อที่ของแผ่นบันทึกไว้เก็บแฟ้มที่เรียงลำดับแล้ว

2. ทำการเรียงข้อมูลตามกำหนดเงื่อนไขแล้วนำระเบียนข้อมูลที่เรียงลำดับแล้วลงในแฟ้มใหม่ในชื่อ test2.t44 (โดยมีข้อมูลส่วนหนึ่งอยู่ในหน่วยความจำของคอมพิวเตอร์) โดยเขียนข้อมูลเรียง

```
PROCEDURE SORTDatabase;
var  SortSTR:array[0..1] of string;
     SortNum:array[0..1] of real;
     Sortck,SortExit:word;   ck:integer; STT:string;
     s,x:word; Sortpercent:real; Mode:byte;

      Procedure SwapData(N:word);
       var No,x1,x2:word; DataField:Char;
       begin
            x1:=(N-1)*SizeOfRecord-1;  x2:=N*SizeOfRecord-1;
          for No:=1 to SizeOfRecord do
             begin
               DataField:=DataDbase[No+x1];
               DataDbase[No+x1]:=DataDbase[No+x2];
               DataDbase[No+x2]:=DataField;
             end;  SortExit:=27;
       end;
begin
     normal(5,22,'Field name : ');  FindField(18,22);
     if TypeOfField[NoField]='N' then Numeric:=true;
     gotoxy(5,23); write('Select Case 1. Ascending 2. Descending ');
        repeat readchr; until ch in ['1','2'];
     normal(5,23,space); Mode:=ord(ch)-48;
     Cursoroff;  Sortck:=CountofRecord-1;
 repeat  SortExit:=0;
         SortPercent:=(CountOfRecord-Sortck)*100/CountOfRecord;
         gotoxy(5,23); writeln(' Sorting  ',SortPercent:2:2,'% ');
   for NoR:=1 to Sortck do
     begin
        FindData(NoR);   SortNum[0]:=DbaseNum;  SortSTR[0]:=DbaseSTR;
        FindData(NoR+1); SortNum[1]:=DbaseNum;  SortSTR[1]:=DbaseSTR;
       case Mode of
        1:begin
           if Numeric then                                    begin
              if SortNum[1]<SortNum[0] then SwapData(NoR);    end
               Else if SortSTR[1]<SortSTR[0] then SwapData(NoR);
          end;
        2:begin
           if Numeric then                                    begin
              if SortNum[1]>SortNum[0] then SwapData(NoR);    end
                Else if SortSTR[1]>SortSTR[0] then SwapData(NoR);
          end;
       end;
     end;
      Dec(Sortck); if SortExit=0 then Sortck:=0;
 until Sortck<=1;
end;
```

รูปที่ 2 ชุดคำสั่งในการเรียงลำดับข้อมูล

ลำดับลงไปตามระเบียนข้อมูลที่เรียงลำดับแล้ว พร้อมกับแสดงบนจอภาพให้ทราบว่าเรียงเสร็จแล้วก็ร้อยละเท่าใด ซึ่งตัวเลขนี้จะหมายรวมถึงที่เรียงลงในแผ่นบันทึกและที่เรียงไว้ในหน่วยความจำรวมกัน

การเขียนไฟล์ test2.t44 นี้มีลักษณะพิเศษต่างจากการเขียนในดอสคือ ดีเบสจะไม่เขียนทับที่ไฟล์ที่ถูกลบด้วยคำสั่ง del หรือ erase ในสารบัญไฟล์ แต่ข้ามไปเขียนที่ส่วนท้ายของสารบัญไฟล์เลย ซึ่งเมื่อได้ทำการเรียงลำดับเสร็จสิ้น ชื่อนี้จะถูกลบออกจากสารบัญไฟล์ทันที โดยจะทำให้เหมือนกับไม่มีการเขียนไฟล์มาก่อน ซึ่งต่างจากการใช้คำสั่ง del, erase ที่ใช้การเปลี่ยนชื่อแรกของไฟล์เป็นรหัสแอสกี้ 229 (ฐาน 10) หรือ E5 (ฐาน 16)

3. เมื่อเรียงเสร็จสิ้นข้อมูลทั้งหมดจะถูกบันทึกลงในไฟล์ test2.w44 แล้วเปลี่ยนชื่อ test2.dbf ก็เป็นการจบการเรียงข้อมูลในดีเบส

นั่นคือถ้าแฟ้มข้อมูลที่นำมาเรียงลำดับมีความยาว 40000 ไบต์ (39.06 KB) ซึ่งไฟล์นี้จะใช้เนื้อที่ในแผ่น 40 กิโลไบต์ จึงต้องมีที่เหลือในแผ่นดิสค์ไม่น้อยกว่า 80 กิโลไบต์ หรือ 81920 ไบต์ จึงจะทำการเรียงลำดับข้อมูลของแฟ้มข้อมูลชุดนี้ได้

การเรียงลำดับที่จะใช้นี้จะใช้การเรียงลำดับแบบที่ง่ายที่สุดซึ่งเป็นแบบเดียวที่ทำได้สำหรับการเรียงข้อมูลที่จัดอยู่ในตัวแปรมิติที่เรียงต่อกันแบบนี้ ขอให้ดูโปรแกรมในรูปที่ 2 ซึ่งมีชุดคำสั่ง SortDatabase

ตัวแปรที่กำหนดไว้คือ SortSTR [0]-[1] สำหรับให้จำข้อมูลของระเบียนที่ติดกันเพื่อนำมาเปรียบเทียบกันในการเรียงลำดับข้อมูลในกรณีที่ชนิดของข้อมูลที่จะเป็นตัวกำหนดการเรียงนั้นเป็นข้อความ และใช้ตัวแปร SortNum[0]-[1] ในกรณีที่เป็นข้อมูลชนิดตัวเลข

นอกนั้นก็เป็นตัวแปรที่กำหนดไว้ใช้งาน ขอให้ศึกษาชุดคำสั่งดังกล่าวนี้ก่อน และขอให้จดเวลาที่ใช้ในการเรียงลำดับว่านานแค่ไหน โดยอาจเขียนเขียนชุดคำสั่งแสดงเวลาก่อนการเรียงลำดับและเวลาเมื่อเรียงเรียบร้อยแล้วแสดงบนจอภาพก็ได้ แต่ไม่ควรให้แสดงเวลาเปลี่ยนไปเรื่อย ๆ เพราะนั้นจะต้องเสียเวลากับการมาแสดงส่วนนี้อีก ในโปรแกรมให้แสดงร้อยละการเรียงลำดับซึ่งเป็นเพียงตัวเลขที่ให้รู้ว่าโปรแกรมกำลังทำงาน

ขออธิบายชุดคำสั่ง swapdata ซึ่งทำหน้าที่สลับข้อมูลของระเบียนที่ตรวจพบว่าต้องเปลี่ยนที่กัน ดังนี้

x1 คือจุดเริ่มต้นของระเบียนข้อมูลแรก ส่วน x2 เป็นจุดเริ่มต้นของระเบียนข้อมูลถัดไป

ให้ทำการสลับข้อมูลจำนวนเท่ากับความยาวของระเบียนข้อมูล โดยกำหนดตัวแปร datafiled มารองรับการสลับที่นี้ ซึ่งจะทำให้ข้อมูลทั้งหมดสลับที่กันได้ และเปลี่ยนค่าตัวแปร Sort Exit:=27;

ชุดคำสั่ง Sortdatabase นั้นทำงานดังนี้

ตรวจหาเขตข้อมูลที่จะใช้เป็นดัชนีในการเรียงลำดับ แล้วตามด้วยการถามว่าจะเรียงแบบใด คือจะเรียงจากน้อยไปหามากหรือจากมากไปหาน้อย

กำหนดตัวแปร Sortck:=Count OfRecord-1;

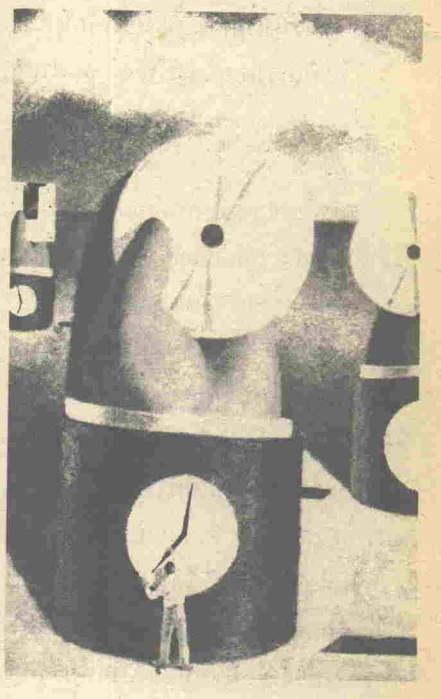
กำหนดตัวแปร SortExit:=0; เพื่อใช้ตรวจว่ามีการสลับที่หรือไม่ การเรียงลำดับจะทำการนำข้อมูลในระเบียนที่ 1 และ 2 มาเปรียบเทียบกัน แล้วเปรียบเทียบข้อมูลที่ 2 กับ 3, 3 กับ 4 ไปเรื่อย ๆ จนกระทั่งถึงระเบียนข้อมูลที่ Sortck

ลดตัวแปร Sortck ลง 1 ค่า หากตัวแปร SortExit=0 ก็จบการเรียงลำดับข้อมูลถ้าไม่เท่าก็วนไปเรียงข้อมูลทั้งหมดนี้ใหม่จนกว่าตัวแปร sortck <=1 ก็จะเป็นการจบการเรียงลำดับข้อมูล

โดยการเรียงนั้นจะทำ 2 กรณีคือ เมื่อข้อมูลเป็นข้อความกรณีหนึ่งและเป็นตัวเลขอีกกรณีหนึ่งซึ่งเงื่อนไขการเรียงลำดับจะต่างกัน

การนับจำนวนข้อมูล

การนับจำนวนข้อมูลอาจกำหนดเงื่อนไขหรือไม่กำหนดก็ได้ ตัวอย่างในชุดคำสั่งที่จะใช้นี้จะแสดงแนวการใช้คำสั่งนับ โดยจะเลือกเขตข้อมูลที่ต้องการให้นับและกำหนดเงื่อนไขของข้อมูล



```
PROCEDURE CountDatabase;
var Count:word; CountSTR:string;
    AllRecord:boolean; ck:integer;
begin
   normal(5,22,'Field name : ');  FindField(18,22);
   normal(6,23,'Condition : '+Copy(Space,1,WidthOfField[NoField])+
               ' Enter for All');   AllRecord:=false;
   Input_STR(18,23,WidthOfField[NoField],Fdec[NoField],NoField);
   CountSTR:=InputSTR; writeln;
   if InputSTR='' then AllRecord:=true;
   If TypeOfField[NoField]='N' then              begin
       Numeric:=true; Val(CountSTR,NumField,ck);  end;
       count:=0;
 if pred(AllRecord) then begin
   for NoR:=1 to CountOfRecord do
     begin
        FindData(NoR);
        if Numeric then  begin
             if (NumField=DbaseNum) then Inc(Count); end
        else
        if (copy(DbaseSTR,1,length(CountSTR))=CountSTR) then Inc(Count);
     end;                    end
       else Count:=CountOfRecord;
     write('         Count : ',count,' records '); readchr;
end;
```

รูปที่ 3 ชุดคำสั่งในการนับจำนวนระเบียนข้อมูลด้วยเงื่อนไขที่กำหนด

ในเขตข้อมูลนั้น ๆ ที่จะให้นับ ซึ่งได้กำหนดไว้ 2 อย่าง คือกำหนดให้ป้อนข้อมูลที่แน่นอนกับการกดแป้น Enter เมื่อต้องการนับทุก ๆ ระเบียนข้อมูล

โปรแกรมอยู่ในรูปที่ 3 ชุดคำสั่งชื่อ CountDatabase ซึ่งขออธิบายสั้น ๆ ดังนี้

หาชื่อเขตข้อมูลที่ต้องการนับโดยเรียกใช้คำสั่ง FindField แล้วตามด้วยการกำหนดเงื่อนไข เมื่อใช้ในการนับ

ถ้าข้อมูลในเขตข้อมูลนั้นเป็นชนิดตัวเลขก็ทำการเปลี่ยนเงื่อนไขเป็นตัวเลขคือ NumField เสียก่อน จากนั้นก็ตรวจว่าเป็นการนับแบบใดถ้านับทั้งหมดตัวแปร Count:=CountOfrecord: ถ้าตรวจตามเงื่อนไขที่ป้อนก็ต้องวนรอบตั้งแต่ต้นแฟ้มข้อมูลไปยังท้ายข้อมูล เพื่อตรวจว่ามีระเบียนข้อมูลใดบ้างที่ตรงกับเงื่อนไขก็ให้นับรวมกับตัวแปร count

เมื่อจบการตรวจก็แสดงจำนวนที่นับได้บนจอภาพ

การหาผลรวมข้อมูล

การรวมข้อมูลทำได้หลาย ๆ แบบ แต่ตัวอย่างในชุดคำสั่งที่ใช้นี้จะแสดงแนวหาผลรวมแบบกำหนดเงื่อนไขได้ 3 แบบ คือ รวมทุกระเบียนข้อมูล, รวมเฉพาะข้อมูลที่ตรงกับเงื่อนไข และรวมเฉพาะข้อมูลที่มีค่าอยู่ในช่วงของเงื่อนไข 2 ค่า

โปรแกรมอยู่ในรูปที่ 4 ชุดคำสั่งชื่อ SumDatabase ซึ่งคล้าย ๆ กับชุดคำสั่ง countdatabase ที่ได้อธิบายไปแล้ว แต่ก็ขออธิบายเพิ่มเติมดังนี้

หาชื่อเขตข้อมูลที่ต้องการนับโดยเรียกใช้คำสั่ง FindField แล้วตามด้วยการกำหนดเงื่อนไขที่หนึ่ง เพื่อใช้ในการรวม ถ้ากดแป้น Enter ก็จะเป็นการรวมทุกระเบียนข้อมูล

จากนี้กำหนดเงื่อนไขที่สอง ถ้ากด Enter ก็จะรวมเฉพาะข้อมูลที่มีค่าเท่ากับเงื่อนไขที่หนึ่ง ถ้าป้อนข้อมูลก็จะรวมเฉพาะข้อมูลที่อยู่ในช่วงของเงื่อนไขที่หนึ่งถึงเงื่อนไขที่สอง

ซึ่งการตรวจสอบในช่วงการวนรอบ for ของตัวแปร NoR ก็จะดำเนินการคล้าย ๆ กับการนับ เพียงแต่เป็นการเพิ่มตัวแปร sum ตามค่าของข้อมูลที่ตรวจพบเท่านั้น

จบลงด้วยการแสดงค่าที่รวมได้นั้นให้ทราบ แต่ถ้าเขตข้อมูลที่ใช้ให้รวมเป็นชนิดข้อความก็จะแจ้งให้ทราบว่าเขตข้อมูลนี้ไม่สามารถนำมารวมกันได้

```
PROCEDURE SumDatabase;
var Sum,Num:real;  ck:integer; AllRecord:boolean;
    Mode;byte;
begin
   normal(5,21,'Field name : ');  FindField(18,21);
   Mode:=0;
if TypeOfField[NoField]='N' then              begin
   normal(5,22,'Condition1 : '+Copy(Space,1,WidthOfField[NoField])+
                        ' Enter for All');
   AllRecord:=false;
   Input_STR(18,22,WidthOfField[NoField],Fdec[NoField],NoField);
       Val(InputSTR,NumField,ck);
       if NumField=0 then AllRecord:=true;
 if pred(AllRecord) then                                    begin
   normal(5,23,'Condition2 : '+Copy(Space,1,WidthOfField[NoField])+
                 ' Condition2>Condition1');
   Input_STR(18,23,WidthOfField[NoField],Fdec[NoField],NoField);
       Val(InputSTR,Num,ck); if Num<>0 then Mode:=1;  end;
       Sum:=0;
   for NoR:=1 to CountOfRecord do       begin
      FindData(NoR);
      if AllRecord then sum:=Sum+DbaseNum
      else  begin
              case mode of
                0:if DbaseNum=NumField then Sum:=Sum+Dbasenum;
                1:if (DbaseNum>=NumField) and (DbaseNum<=Num) then
                                          Sum:=Sum+Dbasenum;
              end;
            end;                                 end;
   write('            Sum : ',sum:0:Fdec[noField],' '); readchr;    end
Else begin
       InputSTR:=cut(NameOfField[Nofield]);
       write('          FIELD : ',INputSTR,'  can'#39't sum ');
       readchr;
     end;
end;
```

รูปที่ 4 ชุดคำสั่งในการหาผลรวมของข้อมูลในระเบียนข้อมูลที่ระบุ

การแก้ไขชื่อเขตข้อมูล

การแก้ไขชื่อเขตข้อมูลนั้นบางทีก็มีความจำเป็นเช่นกัน ในกรณีที่ท่านใช้ปาสคาลตลอดสามารถกำหนดชื่อเขตข้อมูลเป็นภาษาไทยได้แต่จะกลับไปใช้ในดีเบสอีกไม่ได้ เพราะดีเบสนั้นไม่ยอมให้ชื่อเขตข้อมูลรหัสแอสกี้ที่เป็นภาษาไทย ตรงนี้ผู้เขียนคิดว่าเราน่าจะเขียนดีเบสไทย ๆ ที่สามารถรับชื่อเขตข้อมูลเป็นภาษาไทยได้ก็จะน่าสนใจยิ่งขึ้น (ไว้ว่าง ๆ จะลองเสนอแนวคิดมาให้ดูต่อไป)

เพื่อให้การเลือกชื่อเขตข้อมูลเป็นไปอีกแบบหนึ่ง จึงเสนอชุดคำสั่ง Find-Field ซึ่งเป็นชุดคำสั่งที่คล้ายกับชุดคำสั่ง Find Field ที่กล่าวไว้ในตอนที่ 2 เพียงแต่คราวนี้เพิ่มการกลับออกไปเมนูอีกหนึ่งแป้นโดยในโปรแกรมใช้แป้น Home เมื่อกลับออกมาจะได้ตัวแปร NoField ตามลำดับที่ของเขตข้อมูลที่เลือกโดยถ้ากดแป้น Home จะได้ตัวแปรนี้มีค่าเป็นศูนย์

ขอให้ดูชุดคำสั่งในรูปที่ 5 ก่อน การทำงานอย่างคร่าว ๆ คือ กำหนดการทำงานวนรอบ repeat จนกว่าจะกดแป้น Home (funckey=#71)

เลือกชื่อเขตข้อมูลจากชุดคำสั่ง Find-Field ซึ่งจะมีชื่อเขตข้อมูลแสดงบนจอภาพและบอกถึงการใช้แป้นควบคุมให้ใช้แป้นลูกศรขึ้น, ลูกศรลง ในการเลือกชื่อเขตข้อมูล กดแป้น End เมื่อตกลง และกดแป้น Home เมื่อต้องการกลับสู่เมนู

เมื่อตัวแปร NoField มีค่าระหว่าง 1 ถึงจำนวนสูงสุดของเขตข้อมูลให้ทำ

```
PROCEDURE EditFieldName;
var no,n,R:byte;  STT:string;
     Procedure Find_Field;
       begin
           inc(No); if no>CountOfField then no:=1;
           cursorOff; NoField:=0;
           Inverse(19,21,NameOfField[no]);
           Inverse(29,21,' : Move use key '#24#25'   , Select'#26+
                'End   , Menu'#26'Home ');
         Repeat
            readchr;
            case Funckey of
              #72:if No>1 then dec(No) else No:=CountOfField;
              #80:if No<CountOfField then Inc(No) Else No:=1;
            end;
             normal(19,21,copy(Space,1,10));
             Inverse(19,21,NameOfField[no]);
         until Funckey in [#71,#79];
           if Funckey=#79 then NoField:=No;  CursorOn;
       end;

begin   no:=0;
 repeat
   normal(1,22,space+space);
   normal(6,21,'Field name : ');  Find_Field;
  if NoField in [1..CountOfField] then    begin
    repeat
      normal(19,22,space);
      gotoxy(1,22); write(' New Field Name : ');
      Readln(InputSTR);
      for n:=1 to length(InputSTR) do InputSTR[n]:=upcase(InputSTR[n]);
      n:=1; repeat
               STT:=CUT(NameOfField[N]);
               if InputSTR=STT then              begin
                 InputSTR:=''; n:=CountOfField; end;
               inc(n);
            until n>CountOfField;
    until InputSTR[0] in [#1..#10];
    NameOfField[NoField]:=copy(space,1,10);
    for R:=1 to 10 do
     begin
      if R<=Length(InputSTR) then
        begin
          NameOfField[NoField][R]:=InputSTR[R];
          Head[32*NoField+R-1]:=ord(InputSTR[R])
        end
        else  begin
                Head[32*NoField+R-1]:=0;
                NameOfField[NoField][R]:=#0;
              end;
      end;                                    end;
 until Funckey=#71;
end;
```

รูปที่ 5 ชุดคำสั่งในการแก้ไขชื่อของเขตข้อมูล

การเปลี่ยนชื่อเขตข้อมูลนั้นตามคำสั่งในช่วงที่สอง

ให้ป้อนชื่อเขตข้อมูลใหม่ โดยการป้อนนี้จะทำการตรวจชื่อว่าซ้ำกับเขตข้อมูลที่มีอยู่เดิมหรือไม่ถ้าตรงก็จะต้องกลับมาป้อนชื่อเขตข้อมูลใหม่ ตอนนี้จะอยู่ในช่วงวนรอบ repeat ถัดมา ตามคำสั่งนั้นจะจบวนรอบเมื่อความยาวของตัวแปร InputSTR มีค่า 1-10 ในกรณีที่ท่านจะทำงานผ่านปาสคาลตลอดก็สามารถเปลี่ยนชื่อไปเป็นภาษาไทยได้ทุกชื่อเขตข้อมูล แต่จะกลับไปใช้ในดีเบสได้ไม่สมบูรณ์

ช่วงถัดมาคือเป็นการนำข้อมูลที่ป้อนไปใส่ในชื่อเขตข้อมูลที่ต้องการเปลี่ยนโดยต้องเปลี่ยนทั้งชื่อเขตข้อมูลในตัวแปร NameOfField และ ตัวแปร Head ด้วย

การเปลี่ยนแปลงที่เมนูหลัก

เนื่องจากตอนที่ 2 นั้นเราได้เขียนเมนูหลักไว้อย่างสมบูรณ์อยู่แล้ว ตอนที่ 3 นี้เราจึงเพียงแต่เพิ่มเติมและแก้ไขบางจุดเท่านั้นไม่ต้องนั่งคีย์ส่วนนี้ใหม่ทั้งหมด (แก้ไขจากรูปที่ 8 ตอนที่ 2) ดังนี้

แก้ไขค่าคงที่ MenuStr ใหม่จากของเดิมมี 10 ตัวแปรมาเป็น 14 ตัวแปรตามรูปที่ 6 โดยมีค่าคงที่ที่เพิ่มมาคือ Sort, Count, Sum, EditFieldName โดยแทรกเป็นลำดับที่ 10-13

กำหนดค่าตัวแปร MaxMenu จากของเดิม 10 มาเป็น 14 คือ max-menu:=14;

เพิ่มการเลือกกรณีที่ 10-13 ใหม่ตามส่วนท้ายของรูปที่ 6

ส่วนอื่น ๆ ของโปรแกรมทั้งหมดยังเหมือนกับตอนที่ 2 ทุกประการ ซึ่งคิดว่าคงไม่ต้องอธิบายคำสั่งในส่วนนี้ก็คงทำความเข้าใจได้

รูปที่ 6 ส่วนของเมนูที่เปลี่ยนแปลงและเพิ่มเติม

```
PROGRAM ReadDATA_from_dBASE_III;  { use heap and pointer for sorting }
uses crt,dos;
type  VR=^VR_LKS;
      VR_LKS=record
               Head:char;
               Number:array[1..3] of char;
               Name:array[1..16] of char;
               Name2:array[1..16] of char;
               U_nit:array[1..10] of char;
               School:array[1..15] of char;
               Position:array[1..7] of char;
               Level:char;
               District:array[1..10] of char;
               Province:array[1..10] of char;
              end;
var CountOfRecord,SizeOfHead,k,no:word;  f:file;  ch:char;
    VRRecord:array[1..7360] of VR;        Head:array[0..1024] of byte;

PROCEDURE ReadFile;
begin
   assign(f,'VR.DBF');        {$I-}  reset(f,1); {$I+}
 if IOResult<>0 then  begin
     gotoxy(20,21); write('FILE : VR.DTA not found ');
     ch:=readkey;     end
 else            Begin
   blockread(f,head,10); SizeOfHead:=Head[8]+Head[9]*256;
   reset(f,1);           blockread(f,head,SizeOfHead);
   CountOfRecord:=Head[4]+Head[5]*256;
   for k :=1 to CountOfRecord do                          begin
      new(VRRecord[k]);  blockread(f,VRRecord[k]^,89);  end;
   close(f);      end;
end;
```

รูปที่ 6 (ต่อ)

```
PROCEDURE SaveFile(filename:string);
var year,Month,day,minute:word;
begin
   getdate(year,Month,day,minute);
   Head[1]:=year-1900; Head[2]:=Month; Head[3]:=day;
   assign(f,filename);  rewrite(f); reset(f,1);
   blockwrite(f,head,SizeOfHead);
   for k :=1 to CountOfRecord do blockwrite(f,VRRecord[k]^,89);
   close(f);
end;

PROCEDURE Display(Startno,Endno:word);
begin
 if EndNo>CountOfRecord then  EndNo:=CountOfRecord;
 k:=Startno;
 repeat       clrscr;
   writeln(' ============= record no.',k,' ============= ');
   writeln('   NUMBER : ',VRRecord[k]^.number);
   writeln('     NAME : ',VRRecord[k]^.name);
   writeln('  SURNAME : ',VRRecord[k]^.name2);
   writeln('     UNIT : ',VRRecord[k]^.u_nit);
   writeln('   SCHOOL : ',VRRecord[k]^.school);
   writeln(' POSITION : ',VRRecord[k]^.position);
   writeln('    LEVEL : ',VRRecord[k]^.level);
   writeln(' DISTRICT : ',VRRecord[k]^.district);
   writeln(' PROVINCE : ',VRRecord[k]^.province);
   ch:=readkey;  if ch=#27 then k:=EndNo;     inc(k);
 until k>EndNo;
end;

const MenuStr:array[1..14] of string[20]=
        (' UsedBASEFile ',' SavedBASEFile ',' ReadStructure ',
         ' DisplayAll ',' DisplayNext ',' Appenddata ',' Editdata ',
         ' Searchdata ',' Changedata ',' Sort ',' Count ',' Sum ',
         ' EditFieldName ',' Quit ');




      10:SortDataBase;
      11:CountDatabase;
      12:SumDatabase;
      13:EditFieldName;
```

การจัดการแฟ้มข้อมูลแบบเรคอร์ด

ตามที่เขียนมาทั้งหมดได้ดำเนินการแฟ้มข้อมูลแบบกำหนดเป็นตัวแปรชุดเดียว (ตัวแปร Database) ซึ่งทำให้สามารถใช้ได้กับทุกแฟ้มข้อมูลของดีเบส จึงมีความสะดวกในรูปแบบหนึ่ง แต่หากท่านมีโครงสร้างแฟ้มที่คงที่ไม่มีการเปลี่ยนแปลงอีกแล้วก็สามารถมาเขียนโปรแกรมภาษาปาสคาลอีกแบบหนึ่งได้

โดยหากมีการเปลี่ยนโครงสร้างของแฟ้มในภายหลังก็เพียงก็มาแก้ไขหรือเพิ่มเติมบางส่วนเท่านั้น

การดำเนินการแบบนี้คือการสร้างแฟ้มข้อมูลแบบเดียวกับดีเบส โดยกำหนดโครงสร้างของเรคอร์ดให้มีรูปแบบที่เหมือนกับแฟ้มข้อมูลดีเบสที่จะเรียกมาใช้งาน ขอให้ดูได้จากการกำหนดในรูปที่ 7

รูปที่ 7 นี้เป็นโปรแกรมใหม่ซึ่งใช้

```
PROCEDURE Sort;
var sortck:word;  sortpercent:real;
    SortExit:byte;
begin
  clrscr;  Sortck:=CountOfRecord-1;
  writeln('    Sorting ');
 repeat
   SortExit:=0;
   sortpercent:=(CountOfRecord-Sortck)*100/CountOfRecord;
   gotoxy(4,3); write(Sortpercent:5:2,'%  sorted ');
   for  k:=1 to sortck do
      begin
        if VRRecord[k]^.Name>VRRecord[k+1]^.Name then
            begin
               SortExit:=27;
               VRRecord[CountOfRecord+1]:=VRRecord[k];
               VRRecord[k]:=VRRecord[k+1];
               VRRecord[k+1]:=VRRecord[CountOfRecord+1];
            end;
      end;
     dec(sortck); if SortExit=0 then Sortck:=0;
  until sortck<=1;
end;

{=================  SORT dBASE (use heap and pointer) ===============}

begin
   ReadFile;  Display(1,CountOfRecord);
   sort;      Display(1,CountOfRecord);
   SaveFile('VRSORT.DBF');
   ch:=readkey;
end.
```

รูปที่ 7 ชุดคำสั่งการเรียงข้อมูลแบบมีโครงสร้างแบบเรคอร์ด

ชื่อในซอฟต์เมล์ว่า sort.pas ดังที่กล่าวถึงมาครั้งหนึ่งแล้ว เป็นโปรแกรมที่สมบูรณ์ในตัวเองทั้งโปรแกรมไม่ขึ้นกับรูปที่ 1-6 ที่ผ่านมา โดยจะเรียกใช้แฟ้มข้อมูลชื่อ vr.dbf ซึ่งมีโครงสร้างตามที่กำหนดไว้

ขอให้สังเกตโครงสร้างตามชนิดของตัวแปรชื่อ VR_LKS ซึ่งจะประกอบด้วยโครงสร้างดังรูปที่ 8

รวมทั้งระเบียนข้อมูลจะมีความยาว 89 ไบต์เท่ากับโครงสร้างของดีเบส (รูปที่ 1 ตอนที่ 1) โดยการนำข้อมูลทั้งหมดมาเข้าสู่หน่วยความจำในโปรแกรมนี้จะใช้หน่วยความจำที่เรียกว่าฮีพมาช่วยงาน เพราะว่าในหน่วยความจำนี้มีเนื้อที่ใช้งานมากกว่าเพราะมีได้สูงสุดถึง 655360 ไบต์ซึ่งตามโครงสร้างของเรานี้จะใช้ได้กับระเบียนข้อมูลถึง 7,363 ระเบียนข้อมูล (655360/89) ตัวแปรที่กำหนดแบบนี้เรียกว่าข้อมูลแบบไดนามิกส์ (dynamics) ส่วนแบบเดิมนั้นเรียกว่าข้อมูลแบบสแตติกส์ (statics)

การกำหนด VR:=^VR_LKS คือการกำหนดว่าโครงสร้างที่เรากำหนดนี้จะถูกเรียกเข้ามาที่ฮีพ ซึ่งจะต้องใช้คู่กับคำสั่ง new ด้วย

ขอให้ศึกษาโปรแกรมในรูปที่ 7 ก่อน

ขออธิบายโปรแกรมทั้งหมดดังนี้

ชุดคำสั่ง ReadFile จะเริ่มด้วยการกำหนดการเรียกแฟ้มข้อมูลชื่อ vr.dbf พร้อมกับการตรวจว่ามีแฟ้มนี้หรือไม่ ถ้ามีแฟ้มนี้ก็ดำเนินการต่อ โดยอ่านส่วนหัวที่เก็บชื่อเขตข้อมูลเข้ามาเพื่อจะนำเข้าไปเก็บลงในแฟ้มข้อมูลตอนที่เขียนลงไปใหม่

ช่วงการอ่านข้อมูลส่วนที่เป็นระเบียนข้อมูลเข้ามาอยู่ในช่วงวนรอบ for ของตัวแปร k โดยจะอ่านมาทีละ 1 ระเบียนข้อมูลโดยต้องกำหนดตำแหน่งฮีพให้ก่อนจากคำสั่ง new (VRRecord [k]); แล้วจึงอ่านเข้ามาภายหลังด้วยคำสั่ง blockread (f, VRRecord[k] , 89); ซึ่งเป็นรูปแบบของการนำข้อมูลเข้าไปสู่ฮีพ จบลงด้วยการปิดแฟ้มข้อมูล

```
type  VR=^VR_LKS;
VR_LKS=record
       Head:char;   ส่วนคั่นระเบียนข้อมูลซึ่งจะเป็น #32 เสมอ
       Number:array[1..3] of char; เขตข้อมูลที่ 1 ความยาว 3 ไบต์
       Name:array[1..16] of char;  เขตข้อมูลที่ 2 ความยาว 16 ไบต์
       Name2:array[1..16] of char; เขตข้อมูลที่ 3 ความยาว 16 ไบต์
       U_nit:array[1..10] of char; เขตข้อมูลที่ 4 ความยาว 10 ไบต์
       School:array[1..15] of char; เขตข้อมูลที่ 5 ความยาว 15 ไบต์
       Position:array[1..7] of char; เขตข้อมูลที่ 6 ความยาว 7 ไบต์
       Level:char;  เขตข้อมูลที่ 7 ความยาว 1 ไบต์
       District:array[1..10] of char; เขตข้อมูลที่ 8 ความยาว 10 ไบต์
       Province:array[1..10] of char; เขตข้อมูลที่ 9 ความยาว 10 ไบต์
      end;
```

รูปที่ 8 โครงสร้างข้อมูลของ vr.dbf

จะเห็นว่าการดำเนินการตามแนวคิดที่ผู้เขียนเสนอมาทั้งหมดนี้ ไม่มีการเปิดไฟล์ใด ๆ ทิ้งไว้ขณะที่เราจัดการกับข้อมูลเลย

ชุดคำสั่ง Savefile ก็จะคล้ายกัน คือเริ่มด้วยการเปลี่ยนวันเดือนปีของแฟ้มข้อมูลมาตรงกับวันที่เราได้ดำเนินการกับแฟ้มขณะนี้ โดยเขียนแฟ้มขึ้นใหม่ด้วยการนำส่วนหัวที่เก็บรายละเอียดเกี่ยวกับโครงสร้างลงไปก่อน ตามด้วยการเก็บระเบียนข้อมูลตั้งแต่แรกจนสุดท้ายลงไป โดยตามโปรแกรมอยู่ในช่วงวนรอบ for k นั่นเองคือ จะใช้คำสั่ง blockwrite (f,:VRRecord[k]^,89); ซึ่งหมายถึงการนำข้อมูลจากฮีพมาเขียนลงในแฟ้มข้อมูลเรียงลำดับจากต้นจนจบ

ชุดคำสั่ง Display เริ่มด้วยการกำหนดว่าจะแสดงจากระเบียนข้อมูลใดถึงระเบียนข้อมูลใด โดยเราจะเรียกแสดงเพียงบางเขตข้อมูลก็ได้ (ลองเขียนโปรแกรมเอง) แต่ในโปรแกรมนี้ได้แสดงข้อมูลทั้งหมดในแต่ละระเบียนข้อมูล ขอให้สังเกตรูปแบบของการแสดงข้อมูล

ตัวอย่างการแสดงข้อมูลในเขตข้อมูลชื่อ name เราจะใช้คำสั่ง VRRecord [k]^.name เมื่อค่า VRRecord คือตัวแปรแทนระเบียนข้อมูล เช่น VRRecord [5] คือระเบียนข้อมูลที่ 5 (แบบสแตติกส์) ถ้ากำหนดเป็น VRRecord[5]^ ก็จะเป็นตัวแปรไดนามิกส์ในที่นี้เรากำหนดเป็นตัวแปรไดนามิกส์จึงใช้ VRRecord[k]^ ส่วน name คือชื่อเขตข้อมูล ดังนั้น VRRecord[32]^.name ก็คือให้แสดงข้อมูลที่บรรจุอยู่ในเขตข้อมูล name ของระเบียนข้อมูลที่ 32

การแสดงข้อมูลในเขตข้อมูลอื่น ๆ ก็ใช้รูปแบบเดียวกันเพียงแต่เปลี่ยนชื่อเขตข้อมูลเท่านั้น

ชุดคำสั่งเรียงลำดับ sort ตามโปรแกรมนี้แสดงเป็นตัวอย่างเพียงกรณีเดียวคือเรียงด้วยอาศัยเขตข้อมูล name เป็นหลักโดยการเรียงจากน้อยไปหามาก การเรียงลำดับก็ยังใช้แนวคิดเดิมที่กล่าวมาแล้วในตอนต้น ทั้งนี้เมื่อจะให้เปรียบเทียบความไวของการเรียงลำดับว่ามีความแตกต่างกันอย่างไร (โดยบอกให้ทราบก่อนว่าแบบนี้ใช้เวลาน้อยกว่าแบบแรกมากกว่า 18 เท่า) ซึ่งเมื่อลองดูจะรู้ว่าข้อเท็จจริงเป็นอย่างไร

การเรียงลำดับคราวนี้ไวกว่าด้วยเหตุผล 2 ประการ คือ ประการแรกเราไม่ต้องหาข้อมูลที่บรรจุในเขตข้อมูลที่ต้องการนำมาเป็นดัชนีการเรียงลำดับ และประการที่สองการสลับที่ระเบียนข้อมูลทำได้ไวกว่าเพราะสลับทีเดียวทั้งระเบียนข้อมูล (แบบแรกต้องเปลี่ยนทีละไบต์)

ดังนั้นถ้ามีการเรียงข้อมูลการกำหนดแฟ้มข้อมูลแบบนี้จะใช้งานได้สะดวกกว่า เพราะแม้ผู้เขียนจะใช้วิธีการเรียงข้อมูลแบบง่ายที่สุดก็ยังมีความไวสูง ทั้งยังดีกว่าของดีเบสในด้านที่ไม่มีการอ่านหรือเขียนข้อมูลในระหว่างการเรียงลำดับข้อมูลและไม่ต้องมีเนื้อที่ในแผ่นว่างถึง 2 เท่าของความยาวแฟ้มข้อมูล

รายละเอียดการเรียงจะเหมือนกับที่กล่าวมาแล้วในแบบแรกจึงหวังว่าคงทำความเข้าใจได้ไม่ยาก

ส่วนช่วงคำสั่งทำงานจะเริ่มด้วยการอ่านแฟ้มข้อมูล แล้วแสดงผลของข้อมูลที่อ่านมาได้ ตามด้วยการเรียงลำดับข้อมูลและการแสดงข้อมูลที่เรียงลำดับแล้ว จบลงด้วยการเก็บแฟ้มข้อมูลในชื่อ vrsort.dbf

คงต้องจบตอนที่ 3 เพียงแค่นี้ อย่าลืมมีสิ่งใดยังขาดตกบกพร่องหรืออยากให้เขียนถึงก็ส่งไปให้ทราบบ้าง โดยส่งไปที่ ร.ร.ลำปางกัลยาณี อ.เมือง จ.ลำปาง 52000 ทั้งนี้เพื่อให้บทความชุดนี้มีความสมบูรณ์ยิ่ง ๆ ขึ้น

พบกันใหม่คราวหน้า...